Package 'cloudosR'

August 28, 2026

Type Package

Title 'Lifebit' Platform 'API' Client

Version 0.2.4

Description Interacts with the 'Lifebit' Platform Cohort Browser 'API' <https://cloudos.lifebit.ai>. Enables schema discovery, table exploration, and read-only 'SQL' query execution with policy-aware behavior and team-based access control for cohort data analysis. Requires bastion-enabled workspaces for 'API' access.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

Depends R (>= 4.1.0)

Imports httr2 (>= 1.1.0), jsonlite

Suggests testthat (>= 3.1.7), withr, knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.3.3

NeedsCompilation no

Author Leila Mansouri [aut, cre]

Maintainer Leila Mansouri <leila.mansouri@lifebit.ai>

Repository CRAN

Date/Publication 2026-08-28 09:00:02 UTC

Contents

`cloudos.cohort_tables` *List Cohort Tables*

Description

Retrieves the list of available database schemas and tables for a cohort.

Usage

```
cloudos.cohort_tables(profilename = "", cohort_id = "")
```

Arguments

`profilename`	Character. Name of the configured profile to use. If empty or NULL, uses the default profile.
`cohort_id`	Character. ID of the cohort to query schemas for.

Value

List with schema information including databases, tables, and columns.

Examples

```
## Not run:
  # Get available schemas for a cohort
  schemas <- cloudos.cohort_tables(
    profilename = "production",
    cohort_id = "your-cohort-id"
  )

  # Display databases
  cat("Available databases:\n")
  for (db in schemas) {
    cat("  -", db$database, "\n")
  }

## End(Not run)
```

`cloudos.configure` *Configure Lifebit Platform Profile*

Description

Stores API credentials and workspace context for a named profile. This is the required first step before any API wrapper call.

Usage

```
cloudos.configure(
  profilename = "",
  apikey = "",
  workspace_id = "",
  base_url = "https://cloudos.lifebit.ai",
  set_default = FALSE
)
```

Arguments

`profilename`	Character. Name of the profile to create or update.
`apikey`	Character. API key for authentication.
`workspace_id`	Character. Workspace/team ID for API requests.
`base_url`	Character. Base URL for Lifebit Platform API (default: "https://cloudos.lifebit.ai").
`set_default`	Logical. If TRUE, sets this profile as the default (default: FALSE).

Value

Invisible NULL. Prints a success message.

Examples

```
## Not run:
  cloudos.configure(
    profilename = "production",
    apikey = "your-api-key",
    workspace_id = "5c6d3e9bd954e800b23f8c62",
    set_default = TRUE
  )

## End(Not run)
```

cloudos.profile_list *List Lifebit Platform Profiles*

Description

Lists configured profiles so users can confirm available environments.

Usage

```
cloudos.profile_list()
```

Value

Data frame with profile names and metadata (workspace_id, default, created_at, updated_at). Returns empty data frame if no profiles are configured.

Examples

```
## Not run:
  profiles <- cloudos.profile_list()
  print(profiles)

## End(Not run)
```

cloudos.query *Execute SQL Query (Orchestrator)*

Description

High-level function that orchestrates the full query lifecycle: submit -> poll status -> fetch results as dataframe.

Usage

```
cloudos.query(
  profilename = "",
  cohort_id = "",
  sql = "",
  poll_interval = 2,
  max_wait = 600,
  page_size = 1000,
  all_pages = TRUE,
  max_parallel = NULL
)
```

Arguments

profilename	Character. Name of the configured profile to use. If empty or NULL, uses the default profile.
cohort_id	Character. ID of the cohort to query.
sql	Character. SQL query to execute.
poll_interval	Integer. Seconds between status checks (default: 2).
max_wait	Integer. Maximum seconds to wait for completion (default: 600).
page_size	Integer. Number of rows per page (default: 1000).
all_pages	Logical. Fetch all result pages automatically (default: TRUE).
max_parallel	Numeric. Maximum number of HTTP requests issued concurrently when polling task status and fetching page results. Defaults to the cloudosR.max_parallel option, or 10 when that is unset. Lower it to be gentler on the API; set it to 1 for fully sequential behaviour. When TRUE, submits multiple async tasks (one per page) to fetch complete results.

Details

IMPORTANT: Pagination works by submitting separate tasks for each page. When all_pages=TRUE, this function submits multiple async tasks (one per page), waits for all to complete, and combines the results. This may take longer for large result sets.

Status polling and page-result fetching are performed concurrently, with at most max_parallel requests in flight at any moment.

Value

Data frame with query results.

Examples

```
## Not run:
  # Fetch all results
  results <- cloudos.query(
    profilename = "production",
    cohort_id = "699edb4380a6867895f0c9e1",
    sql = "SELECT person_id, gender_concept_id FROM person LIMIT 500"
  )

  # Fetch only first page
  results_page1 <- cloudos.query(
    profilename = "production",
    cohort_id = "699edb4380a6867895f0c9e1",
    sql = "SELECT person_id FROM person",
    all_pages = FALSE,
    page_size = 100
  )

  # Limit concurrency to 4 in-flight requests
  results_gentle <- cloudos.query(
```

```
  profilename = "production",
  cohort_id = "699edb4380a6867895f0c9e1",
  sql = "SELECT person_id FROM person",
  max_parallel = 4
)

## End(Not run)
```

`cloudos.query_count` *Get Total Row Count for a Query*

Description

High-level function that orchestrates the full count lifecycle: submit a count task, poll until complete, and return the total row count. The count endpoint is separate from the data endpoint.

Usage

```
cloudos.query_count(
  profilename = "",
  cohort_id = "",
  sql = "",
  poll_interval = 2,
  max_wait = 600
)
```

Arguments

`profilename`	Character. Name of the configured profile to use. If empty or NULL, uses the default profile.
`cohort_id`	Character. ID of the cohort to query.
`sql`	Character. SQL query whose total row count is requested.
`poll_interval`	Integer. Seconds between status checks (default: 2).
`max_wait`	Integer. Maximum seconds to wait for completion (default: 600).

Value

Integer. Total number of rows matching the query.

Examples

```
## Not run:
  count <- cloudos.query_count(
    profilename = "production",
    cohort_id = "699edb4380a6867895f0c9e1",
    sql = "SELECT person_id FROM person"
  )
```

```
  cat("Total rows:", count, "\n")

## End(Not run)
```

`cloudos.query_count_results` *Fetch Count Results from Async Count Task*

Description

Fetches the total row count from a completed count async task. Use with a task submitted via `cloudos.query_submit_count_async()`.

Usage

```
cloudos.query_count_results(profilename = "", task_id = "")
```

Arguments

`profilename`	Character. Name of the configured profile to use. If empty or NULL, uses the default profile.
`task_id`	Character. Task ID returned from cloudos.query_submit_count_async().

Value

Integer. Total number of rows matching the query.

Examples

```
## Not run:
  task <- cloudos.query_submit_count_async(
    profilename = "production",
    cohort_id = "699edb4380a6867895f0c9e1",
    sql = "SELECT person_id FROM person"
  )
  # ... poll until completed ...
 count <- cloudos.query_count_results(profilename = "production", task_id = task$task_id)

## End(Not run)
```

`cloudos.query_results` *Fetch Async Query Results*

Description

Fetches results from a completed async SQL task and returns as a dataframe.

Usage

```
cloudos.query_results(profilename = "", task_id = "", total_rows = NULL)
```

Arguments

`profilename`	Character. Name of the configured profile to use. If empty or NULL, uses the default profile.
`task_id`	Character. Task ID returned from cloudos.query_submit_async().
`total_rows`	Integer or NULL. Non-negative total row count for the query, used to populate the total_rows/total_pages metadata. Obtain from cloudos.query_count(). Omit to leave them unpopulated (NULL).

Details

Note: Pagination is controlled when submitting the query via cloudos.query_submit_async(), not when fetching results. This function returns whatever page the task was configured for.

The data endpoint does not return a total row count, so `total_rows` and `total_pages` attributes are `NULL` by default. Pass `total_rows` (from `cloudos.query_count()` or `cloudos.query_count_results()`) to populate both. When the response itself carries a `total` field (legacy combined endpoint), it is used as a fallback.

Value

Data frame with query results. Carries attributes: total_rows, page, page_size, total_pages, cursor.

Examples

```
## Not run:
  results <- cloudos.query_results(
    profilename = "production",
    task_id = "69a5c58d626fe626da0025ce"
  )

## End(Not run)
```

cloudos.query_status *Check Async Query Status*

Description

Returns the current status and metadata for a submitted async SQL task.

Usage

```
cloudos.query_status(profilename = "", task_id = "")
```

Arguments

profilename	Character. Name of the configured profile to use. If empty or NULL, uses the default profile.
task_id	Character. Task ID returned from cloudos.query_submit_async().

Value

List with task status, count of results, and other metadata.

Examples

```
## Not run:
  status <- cloudos.query_status(
    profilename = "production",
    task_id = "69a5c58d626fe626da0025ce"
  )
  print(status$status)
  print(status$count_of_results)

## End(Not run)
```

cloudos.query_submit_async *Submit Async SQL Query*

Description

Starts async SQL execution for a cohort and returns a task ID for tracking. Targets the `query-results/data/async` endpoint, which returns rows only (no total row count). Use `cloudos.query_submit_count_async()` or `cloudos.query_count()` to get the total row count separately.

Usage

```
cloudos.query_submit_async(
  profilename = "",
  cohort_id = "",
  sql = "",
  pagination = NULL,
  cursor = NULL
)
```

Arguments

profilename	Character. Name of the configured profile to use. If empty or NULL, uses the default profile.
cohort_id	Character. ID of the cohort to query.
sql	Character. SQL query to execute.
pagination	List (optional). Pagination settings with pageNumber and pageSize. Example: list(pageNumber = 0, pageSize = 100). If NULL, API returns default page.
cursor	Character (optional). Opaque cursor for cursor-based pagination, as returned in a previous page's `attr(result, "cursor")`.

Value

List with task metadata including task_id, status, and full response.

Examples

```
## Not run:
  # Submit query without pagination
  task <- cloudos.query_submit_async(
    profilename = "production",
    cohort_id = "699edb4380a6867895f0c9e1",
    sql = "SELECT person_id FROM person LIMIT 100"
  )

  # Submit query with pagination for page 2 with 50 rows per page
  task <- cloudos.query_submit_async(
    profilename = "production",
    cohort_id = "699edb4380a6867895f0c9e1",
    sql = "SELECT person_id FROM person",
    pagination = list(pageNumber = 2, pageSize = 50)
  )
  print(task$task_id)

## End(Not run)
```

cloudos.query_submit_count_async *Submit Async Count Query*

Description

Starts an async SQL execution that returns only the total row count for a cohort query. Targets the `query-results/count/async` endpoint. Fetch the count with `cloudos.query_count_results()` or use the high-level `cloudos.query_count()`.

Usage

```
cloudos.query_submit_count_async(profilename = "", cohort_id = "", sql = "")
```

Arguments

`profilename`	Character. Name of the configured profile to use. If empty or NULL, uses the default profile.
`cohort_id`	Character. ID of the cohort to query.
`sql`	Character. SQL query whose total row count is requested.

Value

List with task metadata including task_id, status, and full response.

Examples

```
## Not run:
  task <- cloudos.query_submit_count_async(
    profilename = "production",
    cohort_id = "699edb4380a6867895f0c9e1",
    sql = "SELECT person_id FROM person"
  )
  print(task$task_id)

## End(Not run)
```

cloudos.sql_validate *Validate SQL Query*

Description

Validates SQL syntax and references before execution.

Usage

```
cloudos.sql_validate(profilename = "", sql = "")
```

Arguments

profilename	Character. Name of the configured profile to use. If empty or NULL, uses the default profile.
sql	Character. SQL query to validate.

Value

List with validation results including isValid, tableReferences, and columnReferences.

Examples

```
## Not run:
  # Validate a SQL query
  validation <- cloudos.sql_validate(
    profilename = "production",
    sql = "SELECT person_id FROM person WHERE year_of_birth >= 1960"
  )

  if (validation$isValid) {
    cat("SQL is valid\n")
    cat("Tables:", paste(validation$tableReferences, collapse = ", "), "\n")
  } else {
    cat("SQL is invalid:", validation$message, "\n")
  }

## End(Not run)
```

---

print.cloudos_tables *Print method for cloudos_tables*

Description

Print method for cloudos_tables

Usage

```
## S3 method for class 'cloudos_tables'
print(x, ...)
```

Arguments

x	A cloudos_tables object
...	Additional arguments (unused)

Value

The cloudos_tables object x, returned invisibly. Called primarily for its side effect of printing a formatted list of schemas and tables to the console.

Index